Teemu Seppä

Pilvialustapohjainen esineiden Internet

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan ko

Insinöörityö

10.05.2015

Metropolia

Tekijä(t)	Teemu Seppä
Otsikko	Pilvialustapohjainen esineiden Internet
Sivumäärä	33 sivua
Aika	13. toukokuuta 2015
Tutkinto	Insinööri AMK
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Simo Silander IT Arkkitehti Kimmo Kaskikallio

Tässä insinöörityössä perehdyttiin ajankohtaiseen esineiden internetilmiöön ja siihen vahvasti sidoksissa olevaan pilvilaskentaan sekä toteutettiin IBM:n Bluemix-pilvialustassa toimiva esineiden internetsovellusratkaisu, jossa verkkoon liitetyn Android-esineen kiihtyvyysanturin havainnot lähetettiin MqTT-viestintäprotokollaa käyttäen varastoitavaksi Mongo-tietokantaan ja tarjottin käyttäjän saataville verkkoselainkäyttöliittymän kautta.

Työssä perehdyttiin useisiin Bluemixin tarjoamiin tekniikoihin, ja niistä varsinkin Node-Red osoittautui erittäin helpoksi tavaksi nopeasti toteuttaa yksinkertaisia palvelinpuolen toiminnalisuuksia ja rajapintoja. Yksinkertaisten sovellusten kehitys onnistuu graafisen käyttöliittymän avulla vaikka täysin ilman JavaScript-ohjelmointi taitoja.

Työssä kehitetyn sovelluksen osalta jatkokehitettävää jäi ainakin Bluemix IoT -sovelluksen käyttöliittymän osalta. Historiallisen datan esitys ja tarjonta käyttäjälle parantaisivat sovelluksen käyttäjälleen tarjoamaa arvoa huomattavasti. Myös reaaliaikaisen tiedon esitys olisi syytä toteuttaa sulavammin.

Lopuksi todettiin, että kehitysympäristönä Bluemix osoittautui monipuoliseksi ja helppokäyttöiseksi pilvialustaksi, jonka avulla kehittäjän on mahdollista aloittaa sovelluskehitys nopeasti.

Avainsanat	Esineiden Internet, Pilvialusta, Bluemix, Android, MqTT, Node.js, Node-RED

Authors(s) Title Number of Pages Date	Teemu Seppä Cloud based Internet of Things 33 pages 13. toukokuuta 2015
Degree	Bachelor of Engineering
Degree Programme	Bachelor of Software Engineering
Specialisation option	Software Engineering
Instructor(s)	Simo Silander, Senior Lecturer Kimmo Kaskikallio, IT Architect

This Bachelor's thesis explores the topical phenomenon of Internet of Things and related cloud computing. The study was made for IBM Finland using IBM Bluemix cloud platform for software development and Android device as an thing to connect to the Internet.

The main aim of the study was to provide IoT software solution to transfer accelerometer data from Android device to Bluemix using MqTT protocol. The data is then stored in Mongo cloud database and provided to users via web page.

This study also explores the various technologies that Bluemix provides. The graphical programming tool Node-Red turned out to be easy way of implementing simple backend functionality and APIs. Developing simple application can be done even without any JavaScript programming skills.

The developed Bluemix IoT application did leave some room for further user interface development. Providing and presenting more historical data would provide more value to user. Presentation of live data should be also implemented more smoothly.

As an cloud application development platform Bluemix turned out to be versatile and easy to use. It provides fast way for developer to start application developing.

Keywords	Internet of Things, Cloud platform, Bluemix, Android, MqTT, Node.js, Node-RED

Sisältö

1 Johdanto 1

2 Esineiden Internet pilviympäristössä 3

2.1 Esineet verkossa 3
2.2 Pilvipalvelut 6

3 Bluemix IoT -sovelluksessa käytetyt tekniikat 9

3.1 MqTT-viestiprotokolla 9
3.2 IBM Bluemix -sovellusalusta 11
3.3 Internet of Things Foundation 13
3.4 Node-RED 13
3.5 MongoDB 14

4 Android IoT -anturi 16

5 Bluemix IoT -Sovellus 21

5.1 Sovellusalusta 21
5.2 Arkkitehtuuri 22
5.3 Mittaustulosten tallennus pilveen 23
5.4 Käyttöliittymä ja tiedon haku 24

6 Bluemix ja kehitystyökalut 28

6.1 DevOps Services 28
6.2 Node-Red-kehitys 29

7 Yhteenveto 30

Lähteet 32

Käsitteet

Ajax Asynchronous JavaScript And XML on joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia.

Bluemix Avoimen standardin PaaS-pohjainen pilvialusta, pääasiassa sovelluskehitystarkoituksiin.

BSON MongoDB:n käyttämä binääri JSON, eli JSON binääri muodossa

IoT Esineiden internet.

JSON JavaScript Object Notation on avoimen standardin tiedostomuoto, jota käytetään tiedonvälityksessä.

MqTT Tiedonsiirtoprotokolla erityisesti suunnattu laitteiden väliseen viestintään.

Node.js Avoimenlähdekoodin suoritusympäristö, joka mahdollistaa JavaScriptin käytön web- ja palvelinpuolen ohjelmointikielenä.

REST Representational State Transfer. HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen.

RFID Radio Frequency IDentification on radiotaajuinen etätunnistusmenetelmä tiedon etälukuun ja -tallentamiseen käyttäen RFID-tunnisteita eli tageja

IBM Watson Kognitiivisen tietojenkäsittelyn perustuva,ihmisen ajatustoimintaa jäljittelevä keinoäly.

VPN Virtuaalinen erillisverkko, jolla kaksi tai useampia verkkoja voidaan yhdistää julkisen verkon yli muodostaen näennäisesti yksityisen verkon.

1 Johdanto

Tämä insinöörityö tehdään Oy IBM Finland Ab:lle, joka on osa kansainvälistä International Business Machines -teknologia- ja konsultointiyritystä. IBM perustettiin Yhdysvalloissa vuonna 1911 ja Suomeen yritys rantautui 1936. Asiakaskunnan ovat muodostaneet pääsäntöisesti koko yhtiön historian ajan toiset suuret yritykset ja julkisen sektorin toimijat. IBM on aina panostanut innovatiivisuuteen ja hallinnut Yhdysvaltojen patenttitilastojen kärkeä jo 22 vuotta peräkkäin. Työn pohjana käytetäänkin useaa IBM:n tuottamaa sovellus- ja teknologiaratkaisua.

Työn tavoitteena on tutustua ajankohtaiseen esineiden internet (*Internet of Things*)-ilmiöön ja siihen vahvasti sidoksissa olevaan pilvilaskentaan, sekä toteuttaa pilvialustassa toimiva esineiden internet -sovellusratkaisu, jossa verkkoon liitetyn esineen tuottama anturidata varastoidaan pilvialustassa sijaitsevaan tietokantaan ja tarjotaan käyttäjän saataville verkkoselainkäyttöliittymän kautta. Työssä käydään läpi esineiden internet -käsitettä ja sen kehitystä, käyttötarkoitusta sekä tietoturvaa. Päätavoitteena työssä on kuitenkin esineiden internet -anturisovelluksen toteutus pilvipohjaista IBM Bluemix -sovellusalustaa käyttäen.

Työssä esiteltyjen tekniikoiden avulla toteutetaan kokonaisuus, jonka muodostavat Android-sovellus ja Bluemix IoT -sovellus. Sovelluskokonaisuuden tarkoituksena on esitellä verkkoon liitetyn anturin (esineen) mittaustulosten lähetys ja tallennus sekä tarkastella Bluemix-kehitysympäristön soveltuvutta käytettäväksi alati laajenevan esineiden Internet -ilmiön rakennustyökaluna.

Koska valmista esineiden Internet -anturituotetta ei ollut helposti saatavilla, päätettiin verkkoon liitettävä anturi toteuttaa tekemällä Android-puhelimeen sovellus, joka lähettäisi langatonta lähiverkkoa käyttäen puhelimen kiihtyyvyysanturin mittaamat arvot eteenpäin Bluemixin varastoitavaksi. Kiihtyvyysanturi valittiin Android-puhelimen tarjoamien anturien joukosta sen yleisyyden sekä reaalimaailman sovellutuksiensa takia. Tärkeintä oli kuitenkin saada käyttöön jokin anturi, jonka tuloksia voitaisiin esineiden Internet -käyttöön soveltuvalla viestintäprotokollalla välittää verkkoon tallennettavaksi ja hyödyn-

nettäväksi.

Lopuksi työssä tarkastellaan vielä Bluemixin tarjoamia kehitystyökaluja ja pohditaan mahdollisia jatkokehitystarpeita.

2 Esineiden Internet pilviympäristössä

2.1 Esineet verkossa

Internetiä on yleisesti pidetty kanavana, jonka kautta ihmiset jakavat tietoa ja kommunikoivat keskenään. Juuri ihmisten tarve kommunikoida ja jakaa ajatatuksia keskenään johti Internetin kehittämiseen 1980-luvun lopulla. Ympärillemme on muodostunut huomaamattomasti toimiva ja ympäristöönsä sulautuva jokapaikan tietotekniikka (*Ubiquitous computing*), joka mahdollistaa ihmisten sähköisen viestinnän lähes kaikkialla, langattomien verkkojen ja mobiililaitteiden avulla [1]. Esineiden internet (Internet of Things, IoT), jota myös teolliseksi- tai asioiden Internetiksi kutsutaan, laajentaa Internetin infrastruktuurin ja jokapaikan tietotekniikan kattamaan ihmisen lisäksi myös erilaiset laitteet, esineet ja anturit.

Teknologisesti tarkasteltuna esineiden Internet on ollut mahdollista toteuttaa jo kymmenisen vuotta sitten. Kiinteällä tai langattomalla yhteydellä Internetiin liitetyt laitteet ja esineet ovat pystyneet verkon välityksellä jakamaan havaintojaan ympäristöstään ja jakamaan ne ihmisten tai suoraan toisten laitteiden kanssa. Piiri- ja viestintäteknologian jatkuva kehitys on kuitenkin mahdollistanut sen, että verkkoon liittyvät laitteet voivat nykyään olla mitä vain pienistä mittausantureista ja kodin elektroniikkalaitteista aina teollisuudessa käytettäviin työkoneisiin asti. Piirien laskentatehon kasvu ja koon pieneneminen on mahdollistanut uusien algoritmien käyttöönoton datan käsittelyssä erittäin pienissäkin laitteissa.

Myös IPv6-protokollan tarjoaman osoitteavaruuden laajenemisen myötä käyttöön tulevat osoitteet 340 sekstiljoonalle solmulle. Näin ollen on teoreettisesti mahdollista jokaisen maapallon asukkaan kytkeä verkkoon noin 60 000 triljoonaa yksilöllisen verkko osoitteen omaavaa esinettä. Langattomat viestintätekniikat ja niiden kasvavat tiedonsiirtokapasiteetit puolestaan mahdollistavat liikkuvien esineiden, esimerkiksi autojen, liittämisen verkkoon. Langaton viestintä onkin eräs tärkeimmistä tekijöistä esineiden internetin kasvussa ja yleistymisessä . [2]

Suurin hyöty Esineiden Internetissä on se, että se mahdollistaa automatisaation lisäämisen kokonaisvaltaisesti ihmisten arjessa. Teollisuudessa jo olemassa olevaa automatisaatiota voidaan entisestään laajentaa ja tehdä älykkäimmiksi keskenään viestivien laitteiden avulla. Teollisuuslaitteiden valvonta ja huolto voidaan IoT:n avulla pitkälti automatisoida, ja virheen havaitessaan laite osaa itse etsiä mahdollista ratkaisua ongelmaan verkosta.

Teollisuuden jäljessä automatisaatio saapuu myös kotitalouksiin sekä osaksi kaupunkien infrastruktuuria. Rakennuksissa automaatio ja esineiden Internet voidaan liittää osaksi itse rakennusta. Älykkäät rakenteet sisältävät sensoreja, jotka tarkkailevat rakenteiden kosteusarvoja mahdollistaen näin esimerkiksi vesi- tai homevaurioiden ennaltaehkäisyn. Myös talon lämmitys, ilmastointi ja valaistus voidaan liittää verkkoon, mikä tarjoaa asukkaille mahdollisuuden etähallita rakennuksen olosuhteita. Sensorit voidaan valjastaa havainnoimaan ihmisten määrää rakennuksessa ja säätämään lämmitystä tai ilmastointia sen mukaan, onko ketään kotona vai ei.

Kulutuselektroniikan saralla mahdollisuudet ovat lähes ehtymättömät, ja jo nyt markkinoilta löytyy älykkäitä jääkaappeja ja pesukoneita. Myös puettavat laitteet ovat tulleet markkinoille älykellojen ja sykemittarien muodossa. Esineiden Internet mahdollistaa tiedon jakamisen näiden erilaisten laitteiden kesken, mahdollistaen vaikkapa jääkapin valvonnan kellosta käsin. Myöskin ajoneuvoissa olevat useat anturit voidaan valjastaa tarjoamaan mittaustietonsa omistajan mobiililaitteeseen ja varoittamaan mahdollisista vioista. [3]. Myös ajoneuvon etähallinta onnistuu mobiilisovelluksen kautta, ja käyttäjä voi muun muassa hallita ovien lukituksia tai käynnistää auton lämmityksen. Euroopan unionilla on *eCall*-hanke, joka tähtää tulevaisuudessa Euroopan alueella ajoneuvoissa pakolliseen turvajärjestelmään. Onnettomuuden tapahtuessa *eCall*-järjestelmällä varustettu ajoneuvo ottaa automaattisesti yhteyden lähimpään hatäkeskuksee, lähettäen sijainti- ja anturidatansa pelastusvirainomaisille. [4]

Infrastruktuurissa Esineiden Internet näkyy jo tälläkin hetkellä esimerkiksi etäluettavina sähkömittareina, jotka yhdessä sähkönsiirtoverkkojen ja niitä valvovien järjestelmien kanssa muodostavat älykkään sähköverkon, joka puolestaan osaa ohjata ja tasata sähkön kulutusta automaattisesti. Kun esineet pystyvät neuvottelemaan energian käytöstä keskenään, voidaan yllättävät kulutuspiikit estää ja suunnata energia sinne, missä sitä eniten tarvitaan. [5]

Erilaisten laitteiden ja teknologioiden suuresta määrästä johtuen ei vielä ole olemassa mitään standardia, joka määrittelisi, mitä esineiden Internet -käsitteeseen katsotaan kuuluvaksi. Yhteisen viitearkkitehtuurin luomiseksi on kuitenkin tekeillä kansainvälinen *ISO/IEC NP 19654* -standardi, jonka tavoitteena on taata *Plug and Play* -henkinen, yksinkertainen ja saumaton yhteensopivuus laitteiden sekä järjestelmien välillä [6]. Osittain kehitystä hillitsee yritysten varovainen suhtautuminen datan vapauttamiseen yleiseen verkkoon, jossa esineet kommunikoivat ja jakavat tietojaan mahdollisesti toisten yritysten esineiden kanssa. [7]

Teknisenä lähtökohtana esineiden Internetille voidaan pitää sitä, että esineellä pitää olla yksilöllinen tunnus, jolla laite ja sen lähettämä data voidaan linkittää yhteen. Myös yhteys verkkoon luonnollisesti vaaditaan, tapahtuipa se sitten *GSM*, *Wi-Fi*, *Bluetooth*, *RFID*-tunniste tai jotakin muuta tekniikkaa käyttäen. Esineiden Internet käsitteeseen liitetään myös vahvasti se, että laitteen tulee voida havaita tai vaikuttaa ympäristöönsä esimerkiksi lähettämällä havaintodatansa muille tai vastaanottamalla komentoja verkosta ja toimimalla niiden mukaan. Internetiin liitettyjen esineiden ja asioiden yleistyessä sekä niiden integroituessa jokapäiväisiksi toimijoiksi ihmisten rinnalle on kuitenkin todennäköistä, että erillisestä käsitteestä *Internet of Things* luovutaan ja tulevaisuudessa puhutaan vain kaiken internetistä (*Internet of Everything*) tai pelkästään internetistä.

Yhä uusien laitteiden liittyessä verkkoon turvallisen tiedonsiirron ja säilytyksen merkitys korostuu entisestään. Käyttäjän pitää pystyä luottamaan turvallisuuteen ja yksityisyyteen, kun otetaan käyttöön laitteita, jotka käsittelevät arkaluontoista dataa koskien esimerkiksi sijaintia tai terveystietoja. Varsinkin laitteet, joiden toimintojen ohjaaminen onnistuu etänä verkon välityksellä, ovat haavoittuvaisia ja saattavat aiheuttaa suurta vahinkoa väärissä käsissä. Suojaamattomat laitteet ja järjestelmät esimerkiksi terveydenhuolossa, ajoneuvoissa tai kiinteistötekniikkassa, saattavat aiheuttaa jopa hengenvaarallisen uhan. Yhteiskäyttöisyysstandardin puute lisää haavoittuvuutta, kun mahdollisesti joudutaan käyttämään ylimääräisiä välityslaitteita ja yhdyskäytäviä esineiden välillä. Useita jokapäiväisiä laitteita, esimerkiksi jääkaappia, ei ole alunperin suunniteltu julkiseen verkkoon ja näin ollen laitteen suunnittelussa ei välttämättä ole otettu huomioon mahdollisia tietoturvauhkia.

Automaattiset porttiskannerit ja haavoittuvuuksia etsivät skriptit, sekä *Shodanin* kaltai-

set, verkkoon kytkettyjä laitteita listaavat hakukoneet, tekevät tietoturvasta huolehtimisesta entuudestaankin välttämättömämpää. Nämä hakukoneet käyvät itsenäisesti verkon osoiteavaruutta läpi etsien avoimia tai näkyviä reittejä erilaisilla kyselyillä. Kyselyihin laitteet vastaavat tervehdysviestillä, josta yleensä käy ilmi, millainen laite on kyseessä ja jopa laitteen käyttötarkoitus. Esimerkiksi Shodan tulkitsee nämä viestit ja tallentaa julkiseen tietokantaan, jota voidaan käyttää joko selainkäyttöliittymän avulla tai ohjelmallisesti ohjelmointirajapinnan avulla. Laitteen tai järjestelmän näkyvyys verkossa onkin syytä minimoida ja vain toiminnan kannalta välttämättömät portit pitää auki, ja nekin on syytä suojata riittävän vahvalla autentikaatiolla. SSH:n kaltaisten tekniikoiden käyttö auttaa datan suojaamisessa tiedonsiirtotilanteessa, mutta todella arkaluontoinen tieto on syytä suojata myös muilla keinoin, kuten kryptaamalla tai käyttämällä palomuureja tai suojattua VPN -yhteyttä. [8]

2.2 Pilvipalvelut

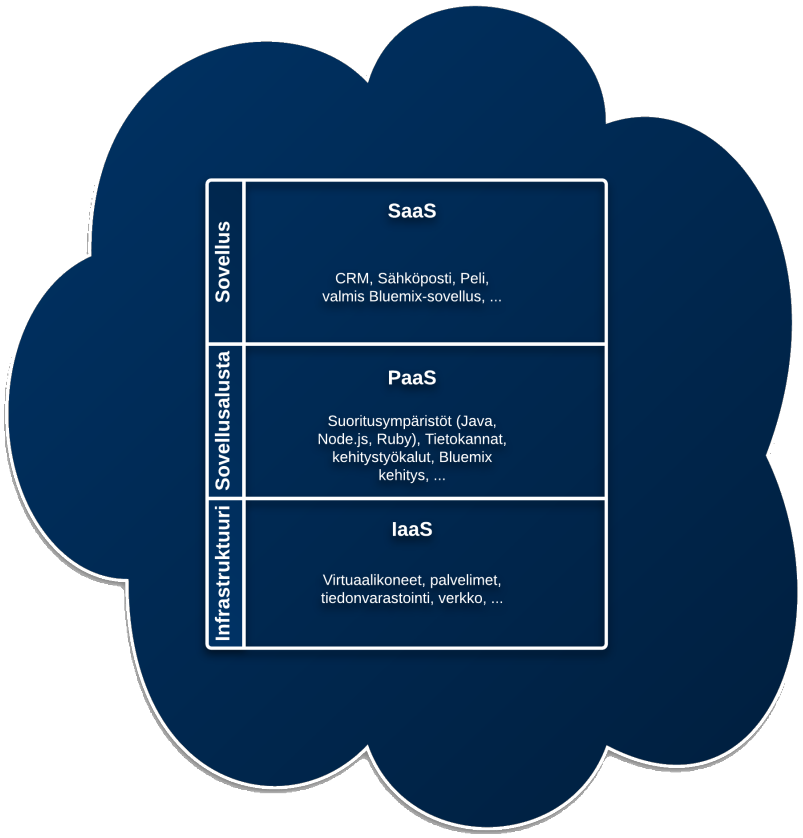
Arvioiden mukaan verkossa viestivien laitteiden määrä ylittää 24 miljardin rajan vuoteen 2020 mennessä. Tämä voimakas lisääntyminen tuo mukanaan haasteita, esimerkiksi verkossa liikkuvan datan määrän kasvun myötä. Tämä huima määrä dataa pitää pystyä välittämään, varastoimaan, prosessoimaan, tulkitsemaan ja esittämään sulavasti sekä mahdollisimman tehokkaasti. Haasteiden myötä avautuu myös mahdollisuus harjoittaa kannattavaa likketoimintaa. On ennustettu, että IoT:n laajeneminen tuottaa uusia markkinamahdollisuuksia eri alojen laajalla rintamalla, aina verkko-operaattoreista ja laitevalmistajista autoteollisuuteen sekä terveydenhuoltoon. Soveltamalla esineiden Internetin tarjoamia ratkaisuja voidaan eri toimialoilla lisäksi saavuttaa toimintojen tehostumista.

Esineiden tuottama valtava määrä dataa vaatii suuren määrän tallennuskapasiteettia sekä resursseja prosessointi- ja verkkolaiteinfrastruktuurilta. Jotta data olisi mahdollista vastaanottaa ja tarjota eteenpäin mahdollisimman monen käyttäjän tai laitteen käyttöön samanaikaisesti ja mahdollisimman realiaikaisesti, on infrastruktuuri kannattavaa toteuttaa hajautetusti pilvipalveluna. Näin saadaan järjestelmään skaalautuvuutta, ja myös kustannukset voidaan jakaa loogisesti käytön mukaan. [2]

Pilvipalvelut ovat palveluja, jotka eivät ole riippuvaisia fyysisestä paikasta ja joita käyte-

tään verkon kautta. Pilvipohjaisessa palvelussa sovellusten ja datan käsittely, eli laskenta, tapahtuu hajautetussa ympäristössä usean fyysisen laitteen avulla. Loppukäyttäjälle pilvipalvelun käyttö ei kuitenkaan eroa paikallisesti tuotetun palvelun käytöstä. Käyttäjälle tarjotut pilvipalvelut saattavat sisältää yksittäisiä sovelluksia, sekä myös kokonaisia infrastruktuureja tai liiketoimintaprosesseja.

Pilvessä tarjottavat palvelut voidaan jakaa kolmeen eri palvelukategiriaan kuten kuvassa 1 on esitetty. Jos asiakkaalle tarjottava palvelu käsittää ainoastaan sovelluksen, käytetään palvelusta nimeä sovellus palveluna *(Software-as-a-Service, SaaS)*. Kun asiakkaalle tarjotaan kokonainen sovellusalusta, jonka päälle asiakas voi kehittää omia sovelluksia, on tämä sovellusalusta palveluna *(Platform-as-a-Service, PaaS*). Asiakkaalle voidaan tarjota myös koko infrastruktuuri palveluna (Infrastructure-as-a-Service, IaaS). *SaaS* on pilviliiketoimintamalleista yksinkertaisin ja rajoitetuin. Tällöin asiakas ulkoistaa vain sovelluksen, ja palveluntarjoaja huolehtii suoritusympärisöistä ja infrastuktuurista. Käyttäjälle suurin hyöty saadaan siitä, ettei sovellusta tarvitse asentaa paikallisesti, vaan se on helposti käytettävissä verkon kautta. Palvelusta veloitetaan useimmin asikasmäärän perusteella.



Kuva 1: Pilvipalvelujen tyypit

Seuraava askel liiketoimintamallin syventämisessä on *PaaS*, jossa asiakkaan vastuulle jää

koko sovellusalusta.*PaaS*-asiakkaan ei kuitenkaan tarvitse huolehtia laiteinfrastruktuurista kuten palvelimista, levytilasta tai verkkoyhteyksistä, mutta hän voi valita useimmissa malleissa suoritusympäristön ja hallita vapaasti sen päällä suoritettavia sovelluksia. *PaaS* luo kehittäjille ikäänkuin hiekkalaatikkoympäristön, jossa erilaisten ratkaisujen testaus on perinteistä sovelluskehitystä nopeampaa ja helpompaa. Parhaimmassa tapauksessa näin sovelluskehitys nopeutuu. *PaaS*:in alla on infrastruktuurikerros IaaS, joka voi olla jopa ulkoistettu kolmannelle osapuolelle *PaaS*-palveluntarjoajan toimesta. Yleisesti integraatio *PaaS*:in *IaaS*:in välilä on toteutettu käyttämällä hyväksi ohjelmointirajapintoja (API), joiden avulla IaaS tarjoaa käytettävissään olevat palvelunsa PaaSin käyttöön.

Pilvipalveluja voidaan tarjota eri muodoissa asiakkaan tarpeiden mukaan. Jos palvelun tavoitteena on olla helposti saatavissa oleva, sekä massoille suunnattu. Se on paras toteuttaa julkisena pilvenä *(public cloud)*. Tällöin pilvipalvelu on kaikkien Internetin käyttäjien saatavilla. Tietoturvasyistä johtuen voi olla tarve rajoittaa pilven käyttäjäkuntaa, ja pitää pilven ja käyttäjän välinen liikenne organisaation sisällä tai jopa pilvilaskennassa käytetyt laitteet organisaation omissa konesaleissa. Tällöin puhutaan ykstyisestä pilvestä *(private cloud)*. Yksityistä pilveä voidaan käyttää myös organisaation sisäisen laitekapasiteetin nopeassa jaossa ja konfiguroinnissa. Julkinen ja yksityinen voidaan myös yhdistää ja luoda yhdistelmäpilvi *(hybrid cloud)*. Tämä voi olla tarpeellista muun muassa lainsäädännöllisistä syistä kuten henkilötietoja käsiteltäessä. *Hybridipilvessä* henkilö- tai muut arkaluontoiset tiedot voidaan pitää yksityisen pilven puolella turvassa, vaikka palvelu toimisikin muuten julkisesti. Tyypillistä pilvipalveluille on se, että niistä veloitetaan käytettyjen resurssejen mukaan. Tämän luo käyttäjälle säästöjä verattuna perinteiin kiinteisiin lisenssimakuihin ja infrastruktuurin ylläpitokustannuksiin. [9]

3 Bluemix IoT -sovelluksessa käytetyt tekniikat

3.1 MqTT-viestiprotokolla

MQTT eli MQ Telemetry Transport on kevyt ja yksinkertainen viestiprotokolla, joka perustuu *julkaise tai tilaa (publish/subscribe)* -arkkitehtuuriin. Se kehitettiin jo vuonna 1999 IBM:n tohtori Andy Stanford-Clarkin ja Arcomin Arlen Nipperin toimesta. Nykyään käytetyinpiin *MqTT*:tä hyödyntäviin sovelluksiin lukeutuu Facebook Messenger -sovellus. MqTT:n tarkoitus on olla avoin ja lisenssimaksuvapaa sekä helposti monille eri alustoille toteutettavissa oleva. Helppokäyttöisyyden luo yksinkertainen komentoviestijärjestelmä sillä MqTT-sovellus voidaan toteuttaa käyttäen vain CONNECT-, PUBLISH-, SUBSCRIBE-, sekä DISCONNECT-viestityyppejä. MqTT on suunniteltu erityisesti laitteille, joissa luotettavan ja nopean tietoliikenneyhteyden muodostaminen voi olla haastavaa. Tekniikan suunnittelussa on tähdätty tiedonsiirtoon vaadittavan kaistan sekä sen prosessointiin tarvittavan laskentatehon minimoimiseen unohtamatta luotettavuutta ja toimitusvarmuutta. Yhtenä pääperiaatteena on, että yksi palvelin pystyy helposti palvelemaan jopa tuhansia asiakaslaitteita. Yhteen yhdistettynä nämä asiat tekevät siitä ihanteellisen tekniikan koneiden välisessä (Machine-to-Machine, M2M) viestinnässä, mikä taas tekee siitä erityisen hyvän esineiden Internet -konseptissa käytettäväksi. Erityisesti erilaiset anturit voivat helposti välittää mittausdataansa MqTT:tä käyttäen. Näin anturin ei tarvitse omata tai tuhlata suuria prosessointiresursseja viestiprotokollien ylläpitämiseksi. Myös mobiiliratkaisut voivat hyötyvät MqTT:n käytöstä sen kaista- ja virrankäyttöystävällisyyden ansiosta.

MQTT:n *julkaise tai tilaa* viestintämallissa tiedon tuottajaa, esimerkiksi mittausanturia, kutsutaan *julkaisijaksi (publisher)* ja tiedon käyttäjää *tilaajaksi (subscriber)*. Tässä välissä on MQTT-palvelin, joka toimii *välittäjänä (broker)*. Tämä mahdollistaa sen, että viestin lähettäjän (sovellus tai laite) ei tarvitse tietää mitään tiedon lopullisesta vastaanottajasta, ei edes sen osoitetta. Viestit julkaistaan *aiheisiin (topic)*, jotka ovat tiettyjä aiheen kattavia kokonaisuuksia. Julkaisija ilmoittaa viestinsä kuuluvan tiettyyn aiheeseen, ja tilaaja voi seurata tai tilata jotain tiettyä aihetta. Asiakkaan *tilaus*, eli haluama aihe voi olla tarkkaan

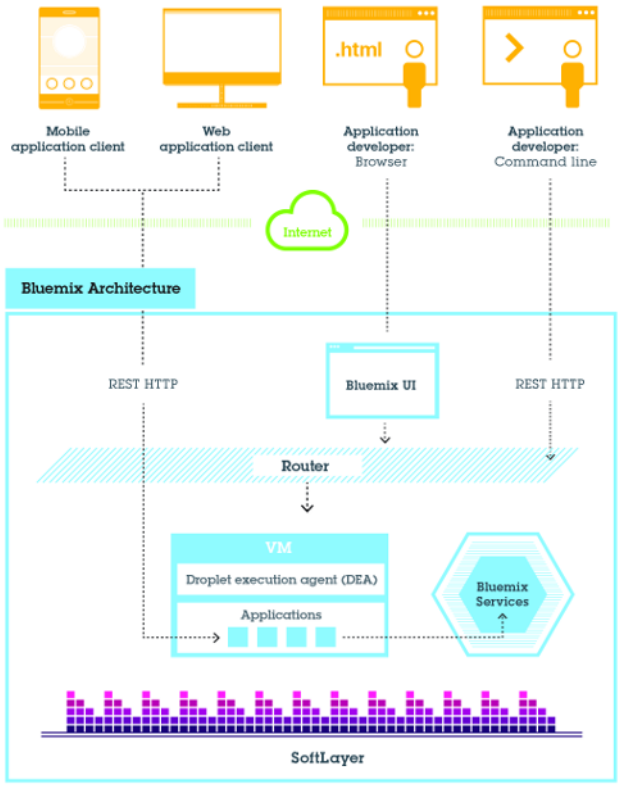
tai löyhästi määritelty. Asiakkaalle voi riittää se, että aihe on halutun kaltainen, tai täysin määritysten mukainen. Kun asiakas lähettää viestin johonkin *aiheeseen* liittyen, kaikki muut, jotka seuraavat tätä *aihetta*, vastaanottavat kyseisen viestin. *Välittäjä*-palvelin myös säilyttää viestit niiden välityksen jälkeen. Kaikki aiheen aiemmat viestit lähetetään myös uusille tilaajille.

Langatonta verkkoa käyttävien laitteiden yleistymisen myötä on lisääntynyt tarve varautua tiedonsiirrossa tapahtuviin häiriöihin. *MqTT*-sovelluksen tiedonsiirron virhekestävyyden määrittelemiseksi on otettu käyttöön kolme erilaista palvelutasoa (*Quality of Service, QoS*): *QoS0, QoS1* ja *QoS2*. Jos palvelutasoksi on määritetty *QoS0*, viesti lähetetään enintään kerran. Tästä johtuen *QoS0* onkin nopein ja kevyin tapa siirtää tietoa *MqTT*:tä käyttäen. *QoS1*-tasolla viesti välitetään niin useasti kuin tarve vaati, mutta aina vähintään kerran. Lähettäjän on tallennettava viesti paikallisesti aina siihen asti, kunnes se vastaanottaa varmistuksen siitä, että viesti on välitetty onnistuneesti. *QoS2*-tasolla käytetään lähettäjän ja vastaanottajan välillä kättelyperiaatetta, jotta voidaan ensin varmistaa luotettava tiedonsiirto ja lähettää viesti tasan kerran, kuten *QoS1*:ssä lähettäjän on tallennettava viesti paikallisesti. *QoS2* on viestin onnistuneen välityksen kannalta turvallisin palvelutaso, mutta samalla myös hitain. Näillä kolmella tasolla pyritään mahdollisimman laajasti kattamaan erinäisten laitteiden ja sovellusten tiedonsiirron virhekestävyys niiden ominaisuuksien ja tarpeiden mukaan.

MqTT-protokollan luotettavuutta lisää myös sisäänrakennettu varautuminen yhteyden katoamiselle. Palvelin saa tiedon, jos asiakkaan yhteys katkeaa poikkeavasti. Tämä mahdollistaa viestin uudelleenlähetyksen tai säilyttämisen myöhempää lähetystä varten. Asiakkaan on myös mahdollista määrittää, kuinka luotettavaa palvelua se haluaa. Jos asiakas on liittyessään asettanut *clean session -lipun* todeksi, kaikki sen tilaukset poistetaan yhteyden katketessa. Jos edellämainittu totuusarvon sisältävä *lippu* on arvoltaan epätosi, yhteyttä pidetään kestävänä, ja tilaukset säilytetään, vaikka asiakas poistuisi tai yhteys katkeaa. Asiakas voi myös määrittää *testamentin (will)* ja kertoa sen palvelimelle. Tämä *testamentti* on viesti, joka julkaistaan sen määrittelemiin aiheisiin, kun asiakas poistuu odottamattomasti. Tämä on erittäin hyödyllinen ominaisuus ja sitä voidaan käyttää esimerkiksi hälyttämään toimintahäiriöstä. [10]

3.2 IBM Bluemix -sovellusalusta

Bluemix on IBM:n avoimen standardin pilvipohjainen sovellusalusta, joka pohjautuu *Cloud Foundry* -nimiseen avoimen lähdekoodiin *PaaS*-ratkaisuun. Kuten kuvassa 2 on esitetty, toimii Bluemixin pohjana *SoftLayer*, joka on IBM:n nykyisin omistama pilvipalveluihin erikoistunut datakeskusinfrastruktuuri. Tämän päällä ajetaan virtuaalisesti Bluemixin suoritusympäristöjä ja palveluja nin sanotuissa säiliöissä (*Container*). Yksi virtuaaliympäristö, joka tunnetaan myös nimellä *Droplet execution agent (DEA)*, voi sisältää monen eri käyttäjän sovelluksia, kukin sovellus omassa säiliössään. Säiliöt on varustettu sovelluksen tarvitsemalla suoritusympäristöllä ja sovelluskehyksellä. Asiakkaat ja kehittäjät käyttävät säiliössä olevia sovelluksia *REST*-rajapintojen kautta.



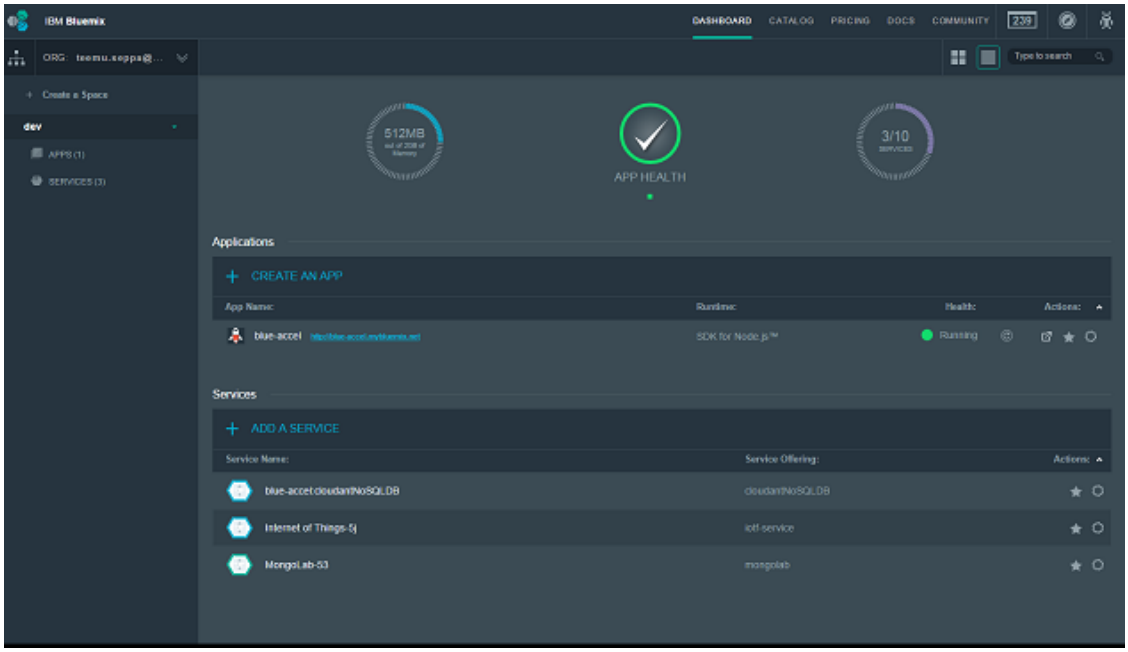
Kuva 2: Bluemix-arkkitehtuuri [11]

Bluemixin tarkoituksena on tarjota sovelluskehittäjille laaja valikoima käyttöönotoltaan helppoja komponentteja pilvipalvelun rakentamiseksi ilman infrastruktuurin hallinnan tuomia huolia. Bluemixin palvelut (*services*) tarjoavat käyttövalmiita toimintoja kehitettävien sovellusten käyttöön. Käyttöönotettu palvelu saa oman instanssin, joka sidotaan kehitettävän sovelluksen instanssiin. Bluemixin palvelu valikoimaan kuuluu palveluja niin IBM:n omasta sovellustarjonnasta kuin kolmansien osapuolten ja avoimen lähdekoodiin

ratkaisuista. Nämä palvelut sisältävät niin tietokanta-, mobiili-, analytiikka- kuin myös *IBM Watsonin* kaltaisia kognitiivisen tietojenkäsittelyn ratkaisuja, jotka pystyvät oppimaan ja keskustelemaan käyttäjän kanssa. Valmiiden vastausten sijasta Watson pystyy itse luomaan vastauksia ja ennusteita, jotka se koostaa laajasta tietomateriaalista.

Tuettuja suoritusympäristöjä ovat tällä hetkellä *Liberty for Java*, *SDK for Node.js*, *Ruby on Rails*, *Ruby Sinatra*, *Go*, *PHP* sekä *Python*. Mahdollista on myös omien *Cloud Foundry* -yhteensopivien suoritusympäristöjen käyttäminen.

Sovellukset viedään Bluemix-ympäristöön joko *Cloud Foundryn cf* -komentorivisovelluksen avulla, tai *Git*-versionhallintaohjelmalla. Eclipse-ohjelmointiympäristöön on myös saatavilla Bluemix-liitännäinen, jolla sovellus voidaan siirtää pilveen. *IBM DevOps Services*-palvelu tarjoaa puolestaan selainpohjaisen käyttöliittymän sovelluksen siirron hallintaan hyödyntämällä *Git*:iä. *DevOps* sisältää *Eclipse Orion* -pohjaisen selainohjelmointiympäristön, jossa koodia voidaan muokata ja tuottaa selaimessa. Bluemixin hallinnointi tapahtuu kuvassa 3 nähtävän selainkäyttöliittymän kautta, tai sitten *cf*-konsolisovelluksen avulla.



Kuva 3: Bluemix-hallintapaneeli

Bluemix on pääsääntöisesti suunnattu sovelluskehitys- ja testaustarkoituksiin IBM:n muiden pilvipalvelujen tarjotessa puitteet kehitettyjen sovellusten varsinaiseen tuotantokäyttöön. Bluemix tarjoaa 30 päivän ilmaisen kokeilujakson, jonka aikana voi veloituksetta kokeilla lähes jokaista palvelua. Testijakson päätyttyä on vielä tarjolla rajoitettu ilmainen käyttö. Muuten veloitus muodostuu pääasiassa käytön mukaan, muutamia kuukausimak-

sullisia palveluja lukuun ottamatta. Käyttäjä voi valita maantieteellisen alueen (*Region*), minkä sisällä sovellusta suoritetaan. Tällä hetkellä valittavana on joko Eurooppa (United Kingdom, Yhdistynyt kuningaskunta) tai Yhdysvaltojen eteläinen alue (US South). Alueen määrittely voi olla käyttäjälle hyvinkin tärkeä ominaisuus esimerkiksi tietosuojalakien vuoksi. Tässä työssä alueena on käytetty US Southia, koska työssä ei käsitellä mitään arkaluontoista dataa. Lisäksi kaikki palvelut eivät vielä ole tarjolla Euroopassa, kuten tämän työn kannalta tärkein palvelu *Internet of Things Foundation*. Alueen valinnassa kannattaa myös pitää mielessä latenssi, joka voi vaikuttaa ratkaisevasti käyttäjäkokemukseen. Region kannattaakin valita sovelluksen oletetun käyttäjäkunnan sijainnin mukaan. [12]

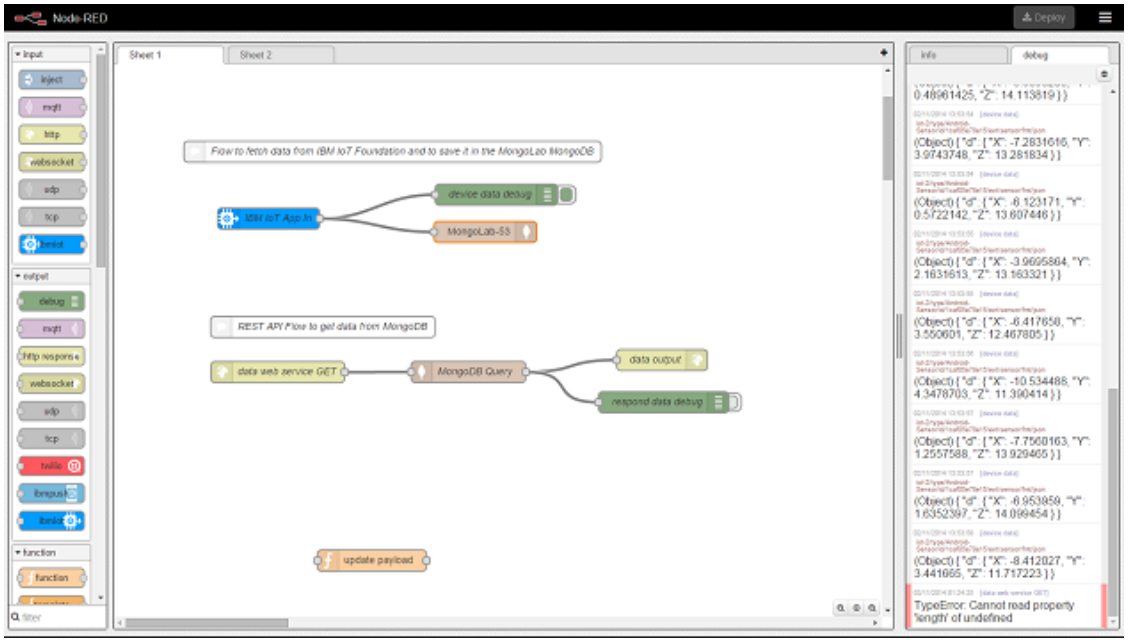
3.3 Internet of Things Foundation

IBM Internet of Things Foundation on pilvipalvelu, joka tarjoaa yksinkertaisen ja helpon tavan yhdistää *IoT*-laiteen tuottama data Bluemix-sovellukseen. Se toimii esineiden Internetin näkökulmasta välittäjänä (*broker*), mutta tarjoaa myös datan reaaliaikaista visualisointia ja varastointia.

3.4 Node-RED

Node-RED on IBM:n kehittämä selainpohjainen visuaalinen avoimen lähdekoodin työkalu, joka on suunniteltu nimenomaan esineiden Internet -konseptiin liittyvää kehitystä varten. Se on rakennettu *Node.js JavaScript* -suorituspäristöä käyttäen. *Node-RED*:issä *JavaScript* funktiot esitetään *JSON*-oliopalikkoina eli solmuina (*node*). Näitä solmuja vedäpudota-tekniikalla lisäilemällä sekä viivoilla yhdistelemällä luodaan sovelluksen *flow*, eli visuaalisesti mallinnetaan datan virtaus solmulta toiselle (kuva 4). Tämän jälkeen sovelluksen vieminen *Node.js*-suoritusympäristöön tapahtuu *deploy*-nappia painamalla. [13]

Node-RED sisältää valmiina varsin kattavan kirjaston erilaisia *nodeja*, mutta niitä voi myös lisätä itse. Lisäksi kirjastoon kuuluu vakiona funktio-*node*, johon on helppo kirjoittaa *JavaScript*-toimintalogiikka esimerkiksi tiedon suodatukseen ja muokkaukseen, tai vaikkapa kokonaisen *html*-sivun luontiin.



Kuva 4: Node-RED-käyttöliittymä, jossa työn IoT-sovelluksen toimintaan tarvittavat flowt

3.5 MongoDB

MongoDB eli *humongous database*, on avoimeen lähdekoodiin perustuva dokumenttiorientoitunut *NoSQL* -tietokanta. Se eroaa muista perinteisistä relaatiotietokannoista siten, että se määrittelee oman kyselykielensä, eikä sillä ole sisäistä relaatiomallia, vaan relaatiot toteutetaan sovellustasolla. Tietokannan tietueet ovat *JSON*:nin kaltaisia olioita, joissa avain-arvo-pareilla määritellään kentän nimi ja arvo. Dokumentin kentät voivat sisältää toisia dokumentteja, taulukkoja tai jopa dokumenttitaulukkoja. *MongoDB*:ssä ei ole perinteisistä tietokannoista tuttua skeemaa, johon taulut kuuluisivat, vaan dokumentit kuuluvat kokoelmaan (*collection*), jonka vastine perinteisissä tietokannoissa on taulu. *Mongo*-titetokantaan tehdyt kyselyt voivat sisältää säännöllisiä lausekkeita tai vaikkapa *JavaScript*-funktioita. [14]

MongoDB soveltuu erinomaisesti pilvialustalle automaattisen skaalautuvuutensa sekä useammalle palvelimelle hajautumisominaisuuksiensa ansiosta. Etuna on myös monistumismekanismi, jolloin sama data voidaan tallentaa useammalle palvelimelle, mikä auttaa varmuuskopioinnissa sekä mahdollisista laitevioista toipumisessa. Tämä mahdollistaa esimerkiksi laitteiden huoltamisen tai vaihdon palvelua keskeyttämättä. Bluemix tarjoaa *Mongo*-tietokannan muodostamiseen kaksi palveluvaihtoehtoa. Suositeltava vaihtoehto, jota myös tässä työssä käytetään, on kolmannen osapuolen, MongoLabin, tarjoama *Mongo-*

Lab-palvelu. *MongoLab* on pilvessä toimiva tietokantapalvelu, joka tarjoaa webpohjaiset hallinta- ja valvontatyökalut sekä tukipalveluja ongelmatilanteissa. Toinen *MongoDB*-palveluvaihtoehto on tavallinen paljas Mongo-tietokanta, jota ajetaan Bluemixissä käyttäjän muun sovelluksen rinnalla. Tämä palvelu on tällä hetkellä luokiteltu kokeelliseksi sen epävakauden ja muuttuvuuden takia. Palvelu saattaa vielä muuttua siten, että vanhemmat versiot eivät ole sen kanssa yhteensopivia. Tätä palvelua ei suositella tuotantokäyttöön.

4 Android IoT -anturi

Esineisen Internet -konseptin perustaksi, eli mittaustiedon lähteeksi valittiin tässä työssä Android-alusta sen verrattain monipuolisen anturitarjonnan ja Android-laitteiden helpon saatavuuden perusteella. Android-laite lähettää anturinsa mittaustulokset *MqTT*-protokollaa käyttäen *Bluemix*iin. Mahdollisista antureista käytettäväksi valittiin kiihtyvyysanturi sen yleisyyden sekä sen tarjoamien reaalimaailman sovellutuksien takia. Kiihtyvyysanturille löytyy käyttötapauksia niin teollisuudesta kuin kuluttajakäytöstäkin. Sitä voidaan käyttää esimerkiksi kokonaisten laitteiden tai erillisten komponenttien kuntoa tarkkailtaessa. Vaarallisen suuret värinät tai muut liikkeet voidaan raportoida eteenpäin, tai niihin voidaan reagoida suoraan esimerkiksi sammuttamalla laite. Näin voidaan ennakoida huollon tarve tai jopa estää komponentin hajoaminen. [15]

Android-laitteille sovellukset kehitetään pääsääntöisesti *Java*-ohjelmointikieltä sekä Android *SDK*:ta (*Software Development Kit*) käyttäen. Sovellus kehitettiin Eclipse-kehitysympäristössä, johon Android *SDK* on liitettävissä. Tässä työssä käytettävän sovelluksen luomiseen käyttettiin lisäksi apuna Fusesourcen *Java MqTT* -asiakas (*client*) -kirjastoa, joka on osa Apache Maven -koontityökalun käyttämää pakettikirjastoa. Mavenia käyttämällä tarvittavien kirjastojen lisääminen projektiin on sangen helppoa. Tässä työssä käytettiin tarjolla olevaa *uberjar*-pakkausta, joka sisältää myös kaikki kirjaston käyttöön vaadittavat riippuvuudet. *Uberjar*-paketti lisätään projektiin *libs*-kansioon, minkä jälkeen *MqTT*-kirjasto on käytettävissä.

Sovellus on toteutettu aktiviteettina (*Activity*), ja se perii *ActionBarActivity*-kirjastoluokan. Sovelluksen käynnistyessä aktiviteetti käynnistyy ja suorittaa *onCreate*-metodin, joka alustaa aktiviteetin ja asettaa käyttöliittymän komponenetit. Jotta Android-laitteen antureja voidaan käyttää sovelluksessa, tarvitaan *android.hardware*-kirjastoluokkia. Anturien arvon muutokset saadaan tapahtumakuuntelija *SensorEventListener*-rajapinnalta, jonka sovellus toteuttaa. Tällöin sovelluksen on myös toteutettava *SensorEventListener*-rajapinnan kaksi metodia *onSensorChanged* ja *onAccuracyChanged*. Metodin *onAccuracyChanged* toteutus on sovelluksessa jätetty tyhjäksi ja keskitytty vain anturien arvojen muutoksiin, jotka saadaan toteuttamalla *onSensorChanged*-metodi, ja antamalla tälle parametriksi, myös

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_mqtt);
    sensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
    lastUpdate = System.currentTimeMillis();
    setupView();
}

@Override
public void onSensorChanged(SensorEvent event) {
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        getAccelometer(event);
    }
}
```

Kuva 5: Android aktiviteettin- ja antureiden alustus

android.hardware-kirjastoon kuuluvan, *SensorEvent*-luokan ilmentymän kuvan 5 mukaisesti. Jos muutoksen aiheuttaneen anturin *SensorEvent*-tyyppi on *ACCELEROMETER*, kutsutaan kuvassa 6 esitettävää *getAccelometer*-metodia. Tämä metodi parsii *SensorEvent*-luokan *event*-muuttujaan tallennetut anturin mittaustulokset *values*-liukulukutaulukkoon. Kiihtyvyysanturi tarjoaa mittaustuloksensa kolmena eri arvona x,y ja z, jotka ovat akselien kiihtyvyyden m/s^2 -komponentit. X-akseli on vaakasuora ja osoittaa oikealle, kun taas Y-akseli on pystysuora ja osoittaa ylöspäin. Z-akseli on syvyysakseli, joka osoittaa laitteen näytöstä ylöspäin. Nämä arvot luetaan liukulukumuuttujiin x,y,z.

Jotta laite ei lähettäisi jatkuvasti dataa pienimmästäkin liikkeestä, on metodiin *getAccelometer* tehty logiikka, jossa mittausarvot lähetetään vasta, kun liike on huomattavan suurta ja edellisestä liikkeestä on kulunut vähintään 200 millisekuntia. Koska maan painovoima ja siitä johtuva putoamiskiihtyvyys vaikuttavat anturin mittauksiin jatkuvasti, on tämä otettava huomioon kiihtyvyyden muutosta tarkkailtaessa. Putoamiskiihtyvyydestä johtuen vaikka laite olisi vain paikallaan tason päällä, anturi havaitsee laitteelle silti kiihtyvyyttä. Toisaalta olessaan vapaapudotuksessa, anturi ei havaitse laitteelle lainkaan kihhtyvyyttä. Jotta mitaaustulokset olisivat mahdollisimman tarkkoja, on painovoiman vaikutus poistettava. Maan painovoima saadaan android.hardware-kirjastoon kuuluvan *SensorManager*-luokan kautta, josta on sovelluksen käynnistyessä *onCreate*-metodissa luotu instanssi. Tässä työssä mittausten tarkkuudella ei kuitenkaan ole kovin suurta merkitystä, ja painovoimaa on käytetty ainoastaan liikkeiden suuruuden suodatuksessa. Riittävän suureksi liikkeeksi on työssä määritelty sellainen liike, jonka kiihtyvyysakselien totaalikiihtyvyyden neliö jaettuna maanpainovoiman neliöllä on suurempi tai yhtä suuri kuin kaksi. Jo yhden

akselin varsin suuri mittauslukema tuo tämän totaalikiihtyvyyden arvon riitävän suureksi ylittämään raja-arvon painovoiman neliöllä jaettaessa. Kun liikkeen suuruus ylittää tämän raja-arvon, parsitaan anturin *x,y* ja *z* sen hetkiset arvot *JSON*-muotoisen *String* olion sisään. Tämän jälkeen kutsutaan *send* metodia, joka lähettää *JSON*-muodossa olevan datan *MqTT*-viestin sisällä eteenpäin määritettyyn kohteeseen.

```
private void getAccelometer(SensorEvent event) {
    float[] values = event.values;
    // Movement
    float x = values[0];
    float y = values[1];
    float z = values[2];

    float accelationSQRT = (x * x + y * y + z * z)
            / (SensorManager.GRAVITY_EARTH * SensorManager.GRAVITY_EARTH);
    long actualTime = event.timestamp;
    if (accelationSQRT >= 2) {
        if (actualTime - lastUpdate < 200) {
            return;
        }

        lastUpdate = actualTime;

        // Sending the message

        sDestination = destinationET;

        String mesValues = "{" + "  \"d\": {" + "\"X\": " + values[0] + ","
                + "\"Y\": " + values[1] + "," + "\"Z\": " + values[2] + "}"
                + " }";

        sMessage = mesValues;

        // allow empty messages
        if (sDestination.equals("")) {
            toast("Destination must be provided");
        } else {

            send();

        }
    }
}
```

Kuva 6: Anturidatan haku ja lähetys

Metodissa *connect* (kuva 7) huolehditaan yhteyden muodostamisesta *broker*-palvelimeen luomalla käyttöönotetun *MqTT*-kirjaston *MQTT*-luokasta ilmentymä ja antamalla sille laitteen yksilöivä *ClientId*, joka *IBM:n Internet Of Things Foundation*:ia käytettäessä on oltava muotoa d: organization-id: type-id: device-id. Tämä asetetaan *setClientId*-metodia käyttämällä. Ensimmäinen parametri *d* kertoo, että *broker*-palvelimeen yhteyden muodostaa laite. Toinen parametri *organization-id* (*org-id*) on sen *IoT Foundationissa* olevan orga-

nisaation tunnus, jolle laite on rekisteröity. *Type-id* on myös *IoT Foundationissa* laitteelle määritelty tyyppi-tunnus, tässä tapauksessa *Android-Sensor*. Viimeinen paramametri on laitteen yksilöivä tunnus, joka tässä tapauksessa on muodostettu laitteen *MAC*-osoitteesta. Yhteyden muodostamiseksi tarvittava *broker*-palvelimen osoite *TCP URI* -skeeman mukaisessa muodossa asetetaan *setHost*-metodilla. Yhdistymisessä käytetään myös käyttäjätunnusta ja salasanaa, jotka asetetaan *setUserName*- ja *setPassword*-metodeilla. Tässä sovelluksessa niin osoite, käyttäjätunnus kuin salasanakin ovat kovakoodattuja arvoja, jotka on saatu laitteen rekisteröintihetkellä *IoT Foundationilta*.

```
private void connect() {
    mqtt = new MQTT();
    mqtt.setClientId("d:[redacted]Android-Sensor[redacted]");

    try {
        mqtt.setHost(sAddress);
        Log.d(TAG, "Address set: " + sAddress);
    } catch (URISyntaxException urise) {
        Log.e(TAG, "URISyntaxException connecting to " + sAddress + " - "
                + urise);
    }

    if (sUserName != null && !sUserName.equals("")) {
        mqtt.setUserName(sUserName);
        Log.d(TAG, "UserName set: [" + sUserName + "]");
    }

    if (sPassword != null && !sPassword.equals("")) {
        mqtt.setPassword(sPassword);
        Log.d(TAG, "Password set: [" + sPassword + "]");
    }

    connection = mqtt.futureConnection();
    progressDialog = ProgressDialog.show(this, "", "Connecting...", true);
    connection.connect().then(onui(new Callback<Void>() {
        public void onSuccess(Void value) {
            connectButton.setEnabled(false);
            progressDialog.dismiss();
            toast("Connected");
        }

        public void onFailure(Throwable e) {
            toast("Problem connecting to host");
            Log.e(TAG, "Exception connecting to " + sAddress + " - " + e);
            progressDialog.dismiss();
        }
    }));

}
```

Kuva 7: Yhteyden muodostaminen *connet*-metodilla

Yhteys luodaan muodostamalla *connection*-niminen olio MqTT-kirjastoluokan *FutureConnection*-rajapintaa käyttäen. *FutureConnection* käyttää pohjanaan *java.util.concurrent.Future-*

rajapintaa, joka käynnistää säikeen suoriutumaan taustalle ja palauttaa tuloksen vasta, kun säie on valmis. Säikeen valmistumista tarkkailemaan on asetettu takaisinkutsumetodi *onui*. Itse varsinaisen *MqTT*-viestin lähetys tapahtuu *send*-metodissa. Ensiksi tarkastetaan yhteys *connect*-metodilla, ja jos yhteyttä ei ole, se luodaan. Kun yhteys on varmistettu, lähetetään viesti *FutureConnection*-ilmentymän *publish*-metodilla. Tälle annetaan parametreiksi kohdeaihe eli *topic*, lähetettävä viesti tavujonoksi muunnettuna sekä palvelunlaatutaso. *IoT Foundationin* käyttö vaati sen, että palvelunlaatutaso *QoS* on sovelluksessa täytynyt asetettaa tasolle 0, eli viestin lähetyksen onnistumisesta ei välitetä ja viesti lähetetään vain kerran.

5 Bluemix IoT -Sovellus

5.1 Sovellusalusta

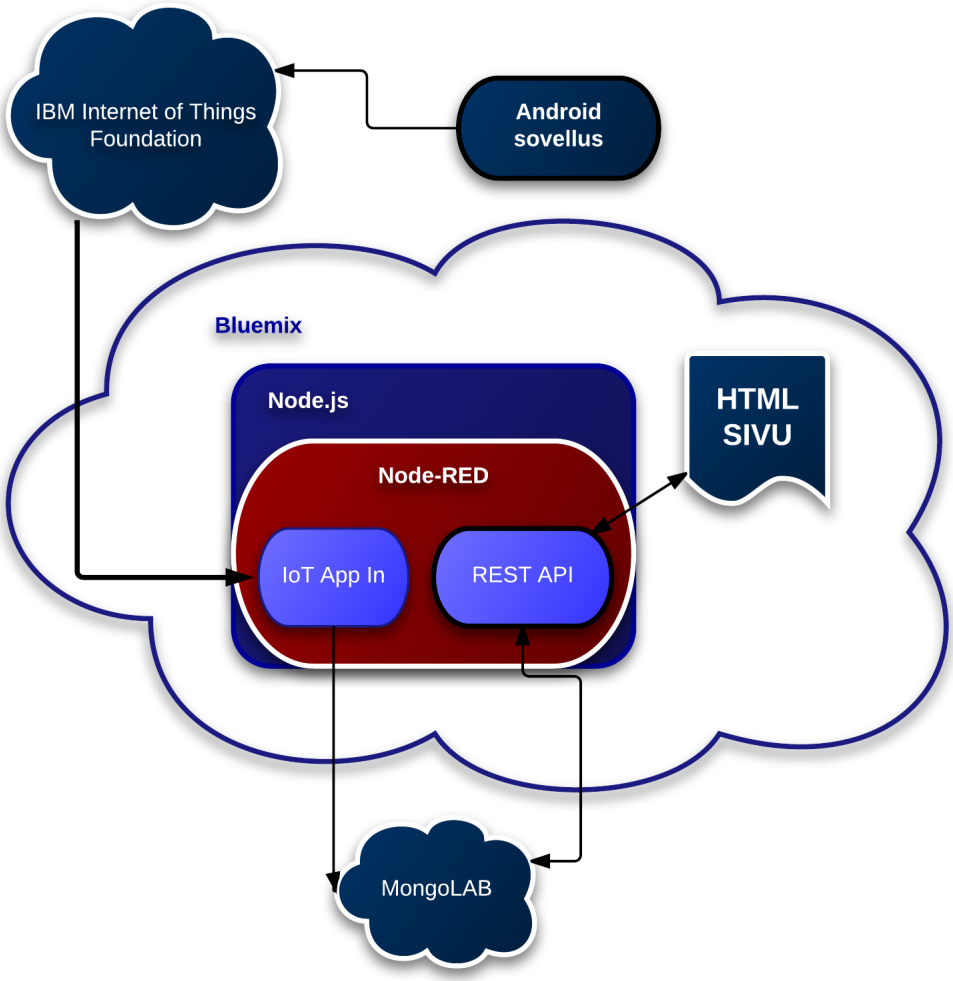
Esineiden Internetin yhdistäminen pilvialustaan on tässä työssä toteutettu käyttäen IBM:n Bluemix-pilvipalvelua. Näin Android-anturin tuottamat mittaustiedot saadaan tarjottua loppukäyttäjän saataville helposti verkon välityksellä. Bluemixiä päädyttiin käyttämään sen tarjoaman IoT-palvelun lisäksi myös siksi, että se on varsin uusi IBM-tuote, johon haluttiin tutustua ja tutkia, miten sovelluskehitys sille tapahtuu. Työn aloitushetkestä Bluemix on kehittynyt todella paljon ja ominaisuuksia on tullut merkkittävästi lisää. Kehitys on pääosin ollut positiivista ja lisännyt helppokäyttöisyyttä ja kehitysmahdollisuuksia uusien ominaisuuksien myötä. Kun työ aloitettiin, oli osa palveluista, kuten *Internet of Things Foundation*, vielä beta- eli esiversiovaiheessa, ja kun siirryttiin valmiiseen *live*-versioon, jouduttiin IoT-sovellus Bluemixissa luomaan uudelleen, sillä uusittu palvelu ei tukenut enää vanhaa sovellusta.

Bluemixin palveluvalikoimasta löytyy valmiita aloituspaketteja *Boilerplateja*, joiden avulla on helppoa ja nopeaa päästä alkuun sovelluskehityksessä. Yksi näistä paketeista on *Internet of Things Foundation Starter*, jota on myös tässä työssä käytetty perustana. Tämä boilerplate asentaa valmiiksi Node.js-suoritusympäristön sekä Node-RED-työkalupaketin ja tarvittavat riippuvuudet. Paketin mukana tulee myös Cloudant NoSQL DB-tietokantapalvelu, jonne on tallennettu Node-RED:in pilvessä tapahtuvaan käyttöön tarvittavaa dataa, kuten valmiit nodet ja esimerkki flow.

Tämän lisäksi toteutettuun IoT-sovellukseen on liitetty IBM Internet of Things Foundation -palvelu, joka toimii Android-sovelluksen lähettämien MqTT-viestien vastaanottajana ja välittäjänä (broker) Bluemixiin. Mittaustulosten tallennukseen olisi voitu myös käyttää Cloudant-tietokantaa, mutta koska Cloudantin laajempi käyttö on osittain maksullista, valittiin tässä työssä käytettäväksi ilmaista Mongo-tietokantaa, Bluemixin kolmansien osapuolten tarjoamiin palveluihin kuuluvan MongoLab-palvelun muodossa.

5.2 Arkkitehtuuri

Bluemix IoT-sovelluksen tehtävänä on säilöä ja tarjota Android-sovelluksen tuottamat mittaustulokset käyttäjän nähtäville. Tiedon varastointi ja analysointi ovat esineiden Internet ilmiössä keskeisellä sijalla. Hyvin toteutetulla sovelluksella voidaan laitteiden tuottamasta datasta luoda käyttäjälle lisäarvoa ja helpottaa jokapäiväistä elämää. Sovelluksen toimintaa ja rakennetta voidaan havainnolistaa sopivan abstraktilla arkkitehtuurikuvauksella.



Kuva 8: Bluemix Internet of Things -sovelluksen arkkitehtuuri

Kuvassa 8 on esitetty Bluemix IoT -sovelluksen toiminta ja sen eri osien väliset kytkökset. Pääsovellusinstanssi sijaitsee Bluemix-pilvipalvelusovellusalustan päällä, ja siihen on liitetty kaksi Bluemixin tarjoamaa palveluinstanssia: IBM Internet of Things Foundation ja MongoLAB. Mittausdata saapuu Android-laitteelta IBM Internet of Things Founda-

tion palveluun *MqTT*-viesteinä. Sieltä ne välitetään MqTT-rajapintaa hyväksikäyttäen Bluemixiin ja tallennetaan *JSON*-formaatissa MongoLAB -palvelun tarjoamaan Mongo-tietokantaan. Tietokannasta mittaustulokset ovat käyttäjän käytettävissä verkkoselaimessa REST-kaltaisen HTTP-rajapinnan kautta.

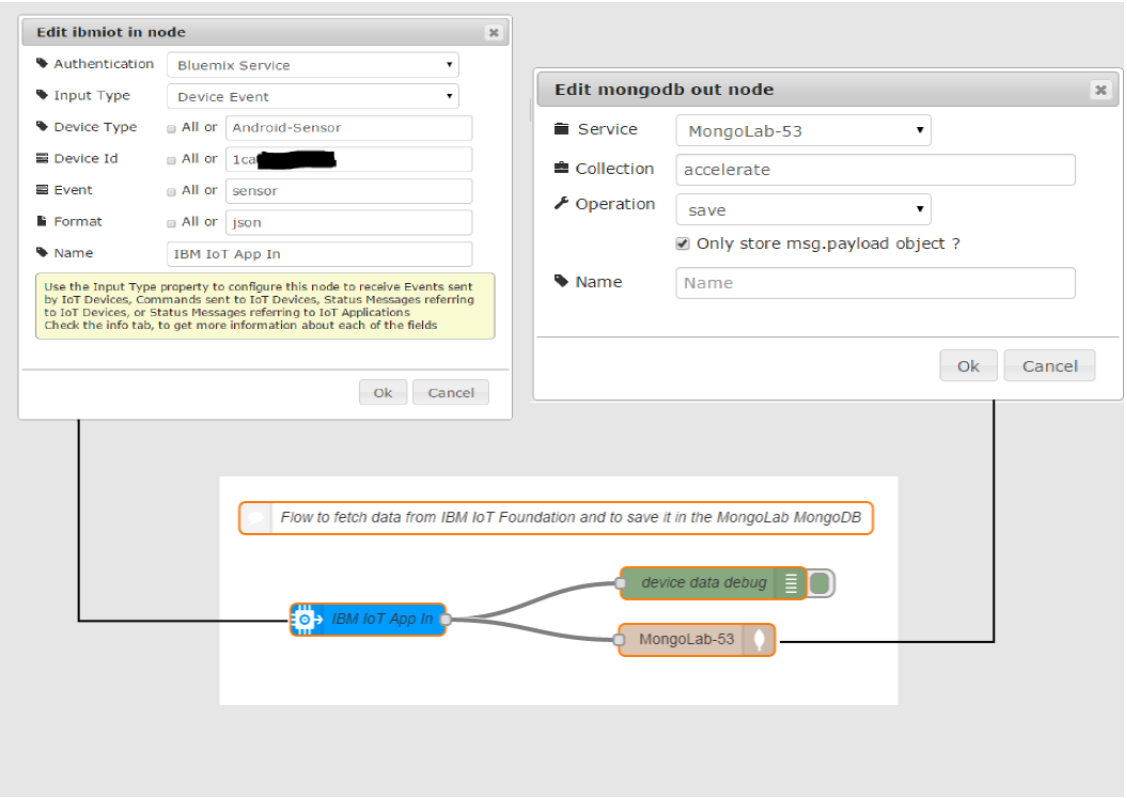
Pääsovellus on toteutettu käyttämällä Node-RED-työkalua, jolla graafisesti selainkäyttöliittymässä luodut *JavaScript*-funktiot suoritetaan Node.js-suoritusympäristössä. Funktiot muodostavat kaksi kokonaisuutta, *flow'ta*, joilla molemmilla on oma tehtvänsä. Ensimmäinen flow suorittaa mittausdatan haun ja tallennuksen tietokantaan ja toinen käyttäjien palvelun vastaanottamalla käyttöliittymältä tulevia *http*-pyyntöjä sekä vastaamalla niihin tietokannasta löytyvällä datalla.

5.3 Mittaustulosten tallennus pilveen

Mittaustulokset tallentava datavirta, flow, on toteutettu käyttämällä kahta noodia (node), eli funktiota, ibmiot-in ja mongodb-out. Ibmiot-in -noodi yhdistää sovelluksen käyttämän instanssin ja IoT Foundation -palvelun. MqTT:n julkaisija-tilaaja mallissa sen roolina tässä sovelluksessa on toimia tilaajana. Kuten kuvasta 9 voidaan huomata, käytetään tunnistautumisessa Bluemix-palvelun käyttöönoton yhteydessä saatuja tunnisteita. Käytännössä tämä tapahtuu niin, että kun IoT Foundation palvelu otetaan ensimmäistä kertaa käyttöön Bluemixin kautta, luo se organisaation IoT Foundationiin, johon palvelu automaattisesti yhdistetään.

Kaikki sovelluksessa käytettyjen palvelujen käyttämät käyttäjänimet, salasanat, osoitteet ja portit tallentuvat Bluemix-sovelluksen *Environment Variables* -asetuksiin. Kun organisaatio on luotu IoT Foundationiin, voidaan sen alle rekisteröidä laitteita. Ilmaiseksi voidaan rekisteröidä 20 aktiivista laitetta sekä 100 MB verkkoliikennettä ja 1 GB tallennustilaa. Yksi organisaatio voi olla ilmaiseksi yhdistettynä enintään 10 sovellukseen. Tässä työssä organisaatiossa on vain yksi laite eli Android-mittauslaite ja noodi on konfiguroitu seuraamaan sen viestejä. Tämä onnistuu asettamalla rekisteröityä laitetta vastaavat arvot laitetyyppi, laitetunniste, tapahtumatyyppi ja viestien muoto sarakkeisiin.

Kun mittaustulosviesti on vastaanotettu, halutaan se tallentaa. Tätä varten on flow'n toiseen päähän liitetty mongodb-out noodi. Kuten IoT Foundationinkin kohdalla, MongoLab-



Kuva 9: Mittaustulosten vastaanotto ja tallennus Node-RED:in avulla

palvelun luonnin yhteydessä luodaan automaattisesti tili- ja tunnistetiedot sekä yhdistetään palvelu sovelluksen instanssiin. Palvelun käyttöönoton jälkeen on MongoLab-palvelun tarjoaman selainkäyttöliittymän kautta luotu tietokantaan kokoelma, eli *Collection*, nimeltään *accelerate*. Tämän kokoelman alaisuuteen kaikki mittaustuloksista muodostuvat tietueet, eli dokumentit, tallennetaan. Kuten kuvassa 9 on esitetty on mongodb-out -noodi konfiguroitu käyttämään Bluemix -palvelua MongoLab-53, joka on sille luonnin yhteydessä anettu nimi, sekä tallentamaan vastaan ottamansa viesti *accelerate*-kokoelmaan.

Tähän tiedontallennusdatavirtakokonaisuuteen eli flow'hun on lisätty myös testaustarkoituksissa debug-noodi, joka on liitetty ibmiot-in-noodiin. Tällöin ibmiot-in-noodin vastaanottamat viesti saadaan tulostetuksi Node-RED:in debug-konsoliin, jolloin laitteen kytkemisen toimivuutta on helppo testata.

5.4 Käyttöliittymä ja tiedon haku

Node-RED soveltuu hyvin datan siirtelyyn ja kevyeen käsittelyyn. Sen avulla rajapintojen luonti on helppoa ja nopeaa. Datan analysointi, visualisointi ja muu prosessointi on

kuitenkin tehokkaampaa suorittaa muilla tekniikoilla. Koska Node-RED-kirjastoa ajetaan Node.js-ajoympäristössä, on siihen helppo liittää Node.js:llä tai JavaScriptillä toteutettuja komponenteja, kuten selainkäyttöliittymä. Node.js:n yleistyessä on perinteinen käyttöliittymän ja palvelinpuolen toiminnallisuuksien ero kaventunut, kun samalla ohjelmointikielellä samassa suoritusympäristössä pystytään toteuttamaan koko sovellus.

```
32
33      <canvas id="smoothie-chart" width="1000" height="400" >
34
35      <script>
36      var seriesX = new TimeSeries();
37      var seriesY = new TimeSeries();
38      var seriesZ = new TimeSeries();
39
40      setInterval(function(){
41          var theUrl = "http://[redacted]-accel.mybluemix.net/accelerate";
42          var xmlHttp = new XMLHttpRequest();
43          xmlHttp.open( "GET", theUrl, false );
44          xmlHttp.send( null );
45
46          var obj = JSON.parse(xmlHttp.responseText);
47          var length = obj.length;
48          var xArray = [length];
49          var yArray = [length];
50          var zArray = [length];
51          var timeArray = [length];
52
53          for(var i = 0; i < length; i++){
54              xArray[i] = obj[i].d.X;
55              yArray[i] = obj[i].d.Y;
56              zArray[i] = obj[i].d.Z;
57              timeArray[i] = new Date(parseInt(obj[i]._id.substring(0,8),16) * 1000);
58          }
59
60          var lastX   = xArray[xArray.length - 1];
61          var lastY   = yArray[yArray.length - 1];
62          var lastZ   = zArray[zArray.length - 1];
63          var lastTime = timeArray[timeArray.length - 1];
64
65          seriesX.append(lastTime, lastX);
66          seriesY.append(lastTime, lastY);
67          seriesZ.append(lastTime, lastZ);
68      }, 2000);
69
70      var chart = new SmoothieChart({millisPerPixel:100,maxValueScale:1.5,scaleSmoothing:1,
71                      grid:{millisPerLine:4000,verticalSections:17},labels:{fontSize:20,precision:6}});
72      chart.addTimeSeries(seriesX, {lineWidth:5,strokeStyle:'#00ff00'});
73      chart.addTimeSeries(seriesY, {lineWidth:5,strokeStyle:'#ff0000 '});
74      chart.addTimeSeries(seriesZ, {lineWidth:5,strokeStyle:'#0000ff'});
75      canvas = document.getElementById('smoothie-chart');
76      chart.streamTo(canvas, 2000);
77
78      </script>
79
```

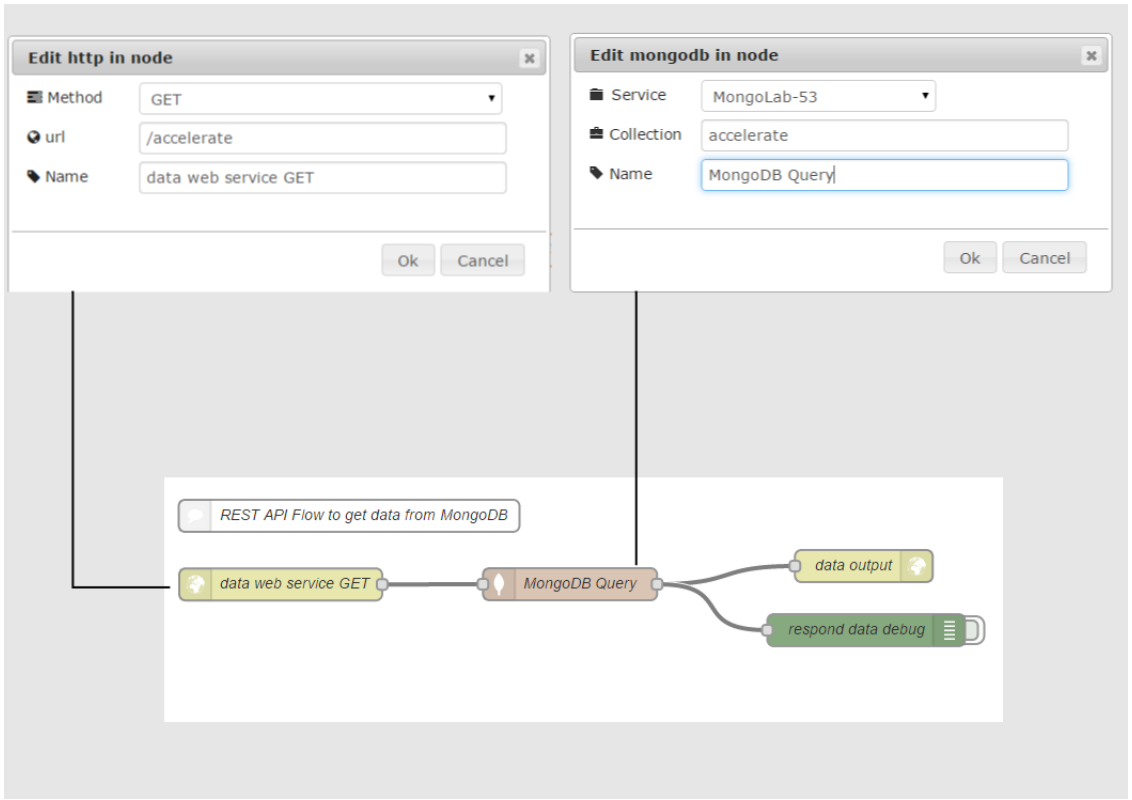
Kuva 10: Käyttöliittymä toiminnallisuus

Bluemix IoT -sovelluksen käyttöliittymä on toteutettu käyttäen moderneja web-kehitystekniikoita, eli *JavaScriptiä* ja *CSS*:ää. Datan haku ja käyttäjälle tarjoaminen on toteutettu *Ajax*-tekniikan mukaisesti koko sivua päivittämättä.

Mittausdatan visualisoinnissa päädyin käyttämään *Smoothie Charts* JavaScript -kirjastoa,

sillä se on kehitetty realiaikaisen datan kuvaamiseen ja täten uusien mittaustulosten liittäminen kuvaajaan on tehty helpoksi. Mittaustulosten arvot kerätään aikasarjaan (*Time series*), joka sitten piirretään korkeamman asteen polynomifunktion kuvaajaa muistuttavalla graafisella kuviolla. Kuten kuvasta 10 ilmenee, ensiksi luodaan kuvaajalle sijoituspaikka HTML:n *canvas*-elementtiä käyttäen. Toiminnallinen logiikka tapahtuu puolestaan JavaScript-skriptissä, jossa ensin alustetaan kiihtyvyyden joka akselille oma Smoothie Charts -aikasarja. Tämän jälkeen *setInterval*-funktiossa haetaan mittaustiedot Mongo-tietokannasta kahden sekunnin välein käyttämällä *XMLHttpRequest*-oliota *GET*-metodilla. Kutsu osoitetaan osoitteeseen, joka koostuu sovelluksen domainista ja päätteestä */accelerate* , johon myös kuvassa 11 näkyvä *http in* -noodi osoittaa.

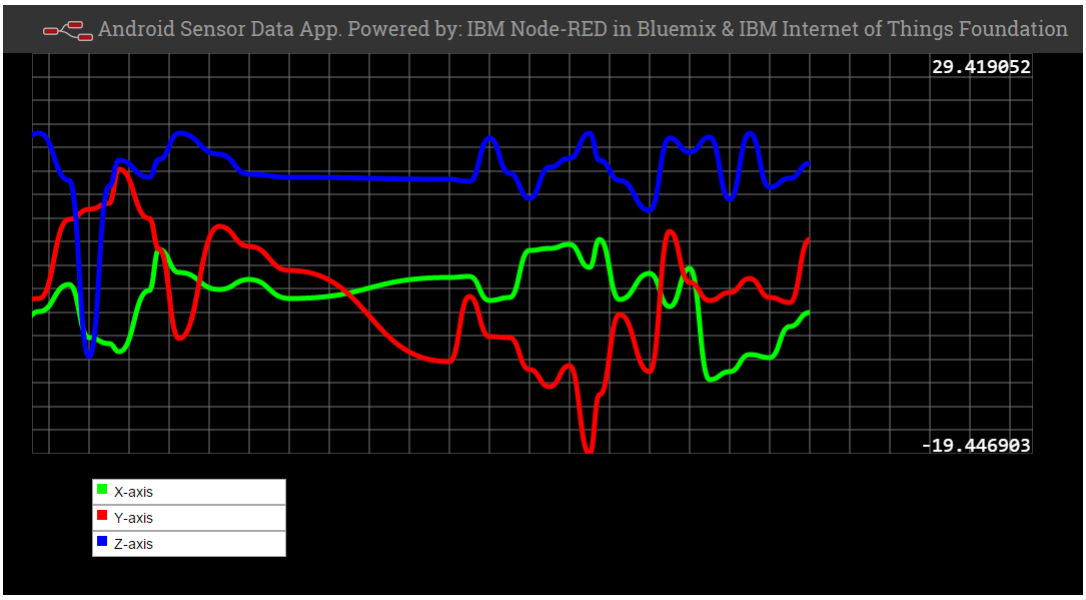
Http in -noodi on osa Node RED:illä toteutettua REST:in kaltaista flowta, jonka tarkoituksena on palvella käyttöliittymästä tulevia kutsuja. Kutsun saadessaan se noutaa kutsua vastaavat tulokset Mongo-tietokannasta ja palauttaa ne */accelerate* osoitteeseeen data output -noodin kautta. Tätä noodia ei tarvitse konfiguroida mitenkään, sillä se palauttaa vastauksen aina sinne, mistä se on http in -noodiin tullutkin.



Kuva 11: Http-pyynnön vastaanotto ja halutun vastauksen palautus Node-RED:illä

Kun XMLHttpRequest-olio on saanut vastauksen, suoritetaan vastaanotetun datan parsinta. Koska data on alunperin lähetetty ja tietokantaan tallennettu JSON-olioina, on sen

parsiminen yksinkertaista. Tietueen pää-oliosta d parsitaan omiin taulukkoihinsa x-,y- ja z-komponentit. Mongo-tietokanta asettaa jokaiselle saapuvalle tietueelle id-tunnisteen, joka on 12 bittiä pitkä BSON, eli binääri json. Tämän tunnisteen alkuosa koostuu aikaleimasta, joka on määritetty tallennushetkestä, ja tätä käytetään hyväksi parsimalla siitä *lastTime* -taulukkoon aina tuoreimman mittaustuloksen saapumisaika. Tätä käytetään puolestaan hyväksi mittaustulosten lisäämisessä visualisoitaviin aikasarjoihin. Lopuksi aikasarjoista muodostetaan kuvan 12 kaltainen kuvaaja, jossa kullekkin aikasarjalle on annettu oma piirtovärinsä.



Kuva 12: Käyttöliittymä ja mittausdata visualisoituna

6 Bluemix ja kehitystyökalut

Kun kehitetään sovellusta Bluemixiä varten, on kehitysmenetelmät syytä valita käytettävän tekniikan mukaan. Esimerkiksi Java-sovellusten kehitykseen Eclipse-ohjelmointiympäristö ja siihen saatavissa oleva Bluemix-liitännäinen tarjoavat toimivan paikallisen kehtiysalustan. Ohjelmointiympäristöistä ei kuitenkaan ole vältämättä apua, jos käytössä on dynaaminen tai löyhästi tyypitetty ohjelmointikieli kuten Pythoni tai Node.js. Jos sovelluksen viennissä käytetään *Cloud Foundry*:n cf-työkalua, voi kehitys tapahtua mitä tahansa työkalua käyttäen.

6.1 DevOps Services

Tässä työssä käytettiin sovelluksen kehityksessä ja sen Bluemixiin viennissä *IBM Bluemix DevOps Services* -palvelua, joka on SaaS-pilvipalveluna toteutettu monipuolinen kehitystyökalu. Sen käyttöönotto tapahtuu helposti Bluemix -selainkäyttöliittymän kautta, jolloin käyttäjälle myös luodaan palveluun *Git*-sovellusvarasto (*repository*), jonka avulla kehitetty sovellus viedään Bluemix:iin. Toimitusketjun hallinassa on mahdollista käyttää myös muita versionhallintatyökaluja tai ulkopuolisia sovellusvarastoja kuten *GitHubia*. Bluemixin kautta luodut sovellusvarastot, eli projektit, ovat oletuksena yksityisiä ja ilmaisia aina, jos kehittäjiä on enintään kolme. Julkiset projektit ovat ilmaisia. *DevOps* tarjoaa versionhallinnan lisäksi myös automaattisen koonti- ja vientityökalun Bluemixiin, joka tosin on osittain maksullinen. Tämän *Delivery Pipeline* -palvelun avulla kehitettävä sovellus kootaan ja viedään automaattisesti Bluemixiin, jokaisen versionhallintapäivityksen myötä.

DevOps sisältää myös verkkoselaimeesa toimivan ohjelmointiympäristön *(Web IDE)*, jota myös tässä työssä käytettiin käyttöliittymän ohjelmoinnissa. Se soveltuu hyvin dynaamisten ohjelmointikielien, kuten JavaScriptin ja Node.js:n kanssa käytettäväksi, jolloin perinteisillä työpöytäsovelluksina toimivilla ohjelmistoympäristöillä ei saavuteta huomattavaa etua. Esimerkiksi suositussa Eclipse-ohjelmistoympäristössä ei oletuksena ole JavaScript-tukea. *Web IDE*:n etu on sen valmiudessa. Sovelluksia voi alkaa kehittämään heti, ilman asennuksia. Web IDE:stä on myös helppo viedä sovellus Bluemixiin nappia painamalla.

Staattisia konfigurointitiedostoja muokatessa muutokset saadaan suoraan voimaan automaattisen *Bluemix Live Sync* -ominaisuuden avulla.

6.2 Node-Red-kehitys

Node-RED -työkalulla kehitys tapahtuu suoraan selaimessa Node-Red-editorissa, ja kun sovelluskehityksen aloittaa käyttämällä Bluemix-aloituspakettia, eli *boilerplatea*, tapahtuu kehitysympäristön pystytys automaattisesti. Node-RED -käyttöliittymä sijaitsee Bluemixissä oletuksena oman Bluemix Node-RED -sovelluksen *URL/red*-osoitteessa. Tämä on syytä suojata salasanalla, jonka asetus onnistuu bluemix-settings.js-tiedostossa määrittelemällä. Salasana asetetaan tietoturvasyistä *MD5*-koosteena.

Sovelluslogiikan muodostaminen tapahtuu yhdistelemällä graafisia palikoita eli noodeja toisiinsa. Valmiita noodeja löytyy useita Bluemixin tarjoaman aloituspaketin mukana, ja kehitys koostuukin lähinnä konfiguraatioasetusten tekemisestä.

7 Yhteenveto

Tämän insinöörityön tavoitteena oli tutkia ajankohtaista esineiden internet -ilmiötä ja siihen vahvasti sidoksissa olevaa pilvilaskentaa. Lisäksi toteutettiin IBM:n Bluemix-pilvialustassa toimiva esineiden Internet-sovellusratkaisu, jossa verkkoon liitetyn Android-esineen kiihtyvyysanturin havainnot lähetettiin MqTT-viestintäprotokollaa käyttäen varastoitavaksi Mongo-tietokantaan ja tarjottin käyttäjän saataville verkkoselainkäyttöliittymän kautta.

Esineiden Internet tarjoaa mahdollisuuden saada jo olemassa olevista arjen esineistä lisäarvoa ja luoda uusia käyttömahdollisuuksia tekemällä niistä älykkäitä. Mahdollisuuksien ohella muodostuu kuitenkin myös tietoturvauhkia ja käyttäjälle pitää pystyä tarjoamaan helposti käytettävä sekä turvallinen tuote. Esineiden tuottama valtava määrä dataa pitää myös pystyä käsittelemään nopeasti ja sulavasti, jonka takia pilvilaskenta ja pilvipalvelut tukevat esineiden Internettiä loistavasti.

Työssä kehitetyn sovelluksen osalta jatkokehitettävää jäi ainakin Bluemix IoT -sovelluksen käyttöliittymän osalta. Historiaallisen datan esitys ja tarjonta käyttäjälle parantaisivat sovelluksen käyttäjälleen tarjoamaa arvoa huomattavasti. Myös realiaikaisen tiedon esitys olisi syytä toteuttaa sulavammin. Muilta osin työssä käytettyjä tekniikoita voidaan täysin käyttää ja työn pääasialliset tavoitteet saatiin täytettyä tyydyttävästi.

Työssä käytetyistä tekniikoista varsinkin Node-Red osoittautui erittäin helpoksi tavaksi nopeasti toteuttaa yksinkertaisia palvelinpuolen toiminnalisuuksia ja rajapintoja. Yksinkertaisten sovellusten kehitys onnistuu grafisen käyttöliittymän avulla vaikka täysin ilman JavaScript-ohjelmointitaitoja.

Kehitysympäristönä Bluemix osoittautui monipuoliseksi ja helppokäyttöiseksi pilvialustaksi, jonka avulla kehittäjän on mahdollista aloittaa sovelluskehitys nopeasti. Bluemix soveltuukin täten hyvin kehitysideoiden nopeisiin kokeiluihin ja demonstroimiseen. Maksuttoman kokeilujakson aikana on mahdollista käyttää ja testata lähes kaikkia lukuisista palveluista ja sen päätyttyäkin rajoitettu käyttö luo mahdollisuuden jatkaa kehitysideoiden kokeilua. Myös Bluemixin tarjoamat DevOps -palvelut osoittautuivat toimiviksi ja no-

peiksi käyttää. Sovelluksen käyttämän tietomäärän kasvaessa on kuitenkin turvauduttava maksullisiin palveluihin, sillä tallennustilaa sisältyy maksuttomaan käyttöön vähän.

Lähteet

1 Jokapaikan tietotekniikka. wikipedia.org; 2013. Saatavilla osoitteesta: http://fi.wikipedia.org/wiki/Jokapaikan_tietotekniikka [viitattu 29.12.2014].

2 Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems; 2013. Saatavilla osoitteesta: http://www.sciencedirect.com/science/article/pii/S0167739X13000241 [viitattu 15.01.2015].

3 The Internet Of Things Is A Standards Thing. electronicdesign.com; 2014. Saatavilla osoitteesta: http://electronicdesign.com/communications/internet-things-standards-thing [viitattu 02.03.2015].

4 eCall. www.imobilitysupport.eu; 2015. Saatavilla osoitteesta: http://www.imobilitysupport.eu/about-ecall [viitattu 09.03.2015].

5 What the Internet of Things Will Mean for the Smart Grid. smartgrid.ieee.org; 2011. Saatavilla osoitteesta: http://smartgrid.ieee.org/june-2011/95-what-the-internet-of-things-will-mean-for-the-smart-grid [viitattu 04.03.2015].

6 Esineiden internetin viitearkkitehtuurista standardi. sfs.fi; 2014. Saatavilla osoitteesta: http://www.sfs.fi/standardien_laadinta/sfs_n_tekniset_komiteat_ja_seurantaryhmat/it-standardisointi/it_-_ajankohtaista/esineiden_internetin_viitearkkitehtuurista_standardi.2020.news [viitattu 08.11.2014].

7 Teollinen internet – uhka vai mahdollisuus? wikipedia.org; 2014. Saatavilla osoitteesta: http://castrenblog.com/2014/09/30/teollinen-internet-uhka-vai-mahdollisuus/ [viitattu 3.5.2015].

8 Tommi Tiitinen. Tietoturvallisuuden haasteet Internetiin kytketyssä teollisessa automaatiojärjestelmässä. Suomi; 2014.

9 Cloud computing. wikipedia.org; 2014. Saatavilla osoitteesta: http://en.wikipedia.org/wiki/Cloud_computing [viitattu 08.11.2014].

10 Valerie Lampkin, Weng Tat Leong, Leonardo Olivera, Sweta Rawat, Nagesh Subrahmanyam, Rong Xiang, et al. Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry. Yhdysvallat: IBM Redbooks; 2012.

11 Bluemix architecture. www.ng.bluemix.net; 2014. Saatavilla osoitteesta: https://www.ng.bluemix.net/docs/#overview/overview.html#ov_arch [viitattu 26.01.2015].

12 Bluemix overview. bluemix.net; 2014. Saatavilla osoitteesta: https://www.ng.bluemix.net/docs/#overview/overview.html#overview [viitattu 10.11.2014].

13 What is Node-Red? learn.adafruit.com; 2014. Saatavilla osoitteesta: https://learn.adafruit.com/raspberry-pi-hosting-node-red/what-is-node-red [viitattu 13.11.2014].

14 Introduction to MongoDB. http://docs.mongodb.org/; 2014. Saatavilla osoitteesta: http://docs.mongodb.org/manual/core/introduction/ [viitattu 13.11.2014].

15 Accelerometer. wikipedia.org; 2014. Saatavilla osoitteesta: http://en.wikipedia.org/wiki/Accelerometer [viitattu 29.12.2014].